May 19, 2017

Advanced Individual Cyclist Experience Mapping: Database Design and Implementation

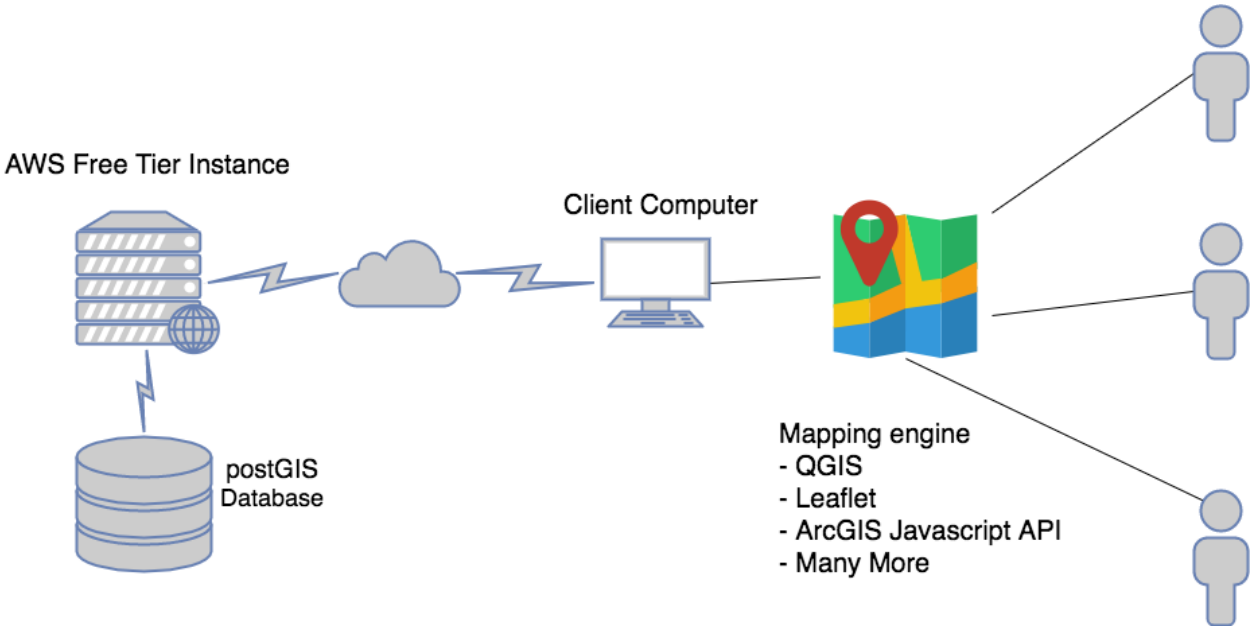
Background:

Metropolitan cities have recently placed emphasized focus on the transition from vehicle dominated traffic networks to multimodal networks. The push for a sustainable city can have many positive impacts within the communities of the overall city, creating ‘‘urban environments which are safer, more sociable and less environmentally damaging’’ (Tight et al., 2011, p. 1580). Many cities throughout the US, such as New York, Washington DC, and Baltimore, have begun converting street space that was once dedicated to parking and vehicular traffic into cycling paths.

Recent scientific literature (Duppen and Spierings, 2013; Snizek et al. 2013) have studied bike commuters and the impact these infrastructure changes have had on them, in order to inquire how cities can meet their needs, and thus help the city attract more bike commuters. During a bicycle commute, the rider is able to connect with the city at large through their senses of touch, smell, and sound. (Duppen and Spierings, 2013). Each rider’s route is considered to be a unique ‘sensescape’, known as the relationship between the urban environment and the sensory body (Degan, 2002), which impacts their outlook on the day at work and home.

Being a relatively new concept, personal biking data is not available through city open data portals, and thus data would have to be collected through a survey of bike commuters within a metropolitan area. This project was designed to provide insight into the creation of a distributable GIS database that can be used to create an online survey for biking individual experiences within a metropolitan area. The intent is to provide steps and guidelines into the installation and deployment of a spatial database instance, creation of the a database schema and tables for the biking survey, and introduction of open source database and GIS tools that allows anyone to recreate the database free of charge. The resulting technology can be used to create an advanced model for the prediction of cyclist’s experiences within new and existing infrastructure.

Stack Design:



The above diagram outlines the proposed stack design using Amazon Web Services (AWS) and a postGIS database. AWS is a cloud services platform that can provide secure computing space for server development and implementation. With AWS, a full spatial database stack can be deployed allowing multiple users to view and edit geospatial data. PostGIS was chosen for this stack because it is an open source spatial database extension that is built on top of postgreSQL, a widely used and supported SQL database software.

In the diagram, the AWS instance is able to interact with the postGIS database which is also hosted on AWS. The instance is configured with specific ports open, allowing users to connect to the database from the world wide web. The cloud in the diagram indicates the addition of web services that can be added to the instance, allowing for specialized queries and edits to be performed on the database. These services could be developed using an HTTP server such as apache or node.js. QGIS and ArcMap are able to connect the the postGIS instance and create spatial features with absolutely no backend scripting necessary.

Database Installation:

This project used a free tier AWS EC2 linux Ubuntu instance. This product allows the owner to connect to a virtual machine in Amazon's cloud through a terminal command line. Once connected the owner is able to install programs and edit files through the terminal command line. To connect to the EC2 instance the user must have a private key provided by Amazon and the hostname of the instance. The follow command can connect the owner to the instance-

```
ssh -i "AWSTesting.pem" ubuntu@ec2-34-209-200-70.us-west-2.compute.amazonaws.com

Note: The .pem file must be in the current directory of the terminal window or the path the
the .pem file must be specified, when connecting.
```

This command connects to ubuntu@ec2-34-209-200-70.us-west-2.compute.amazonaws.com through "'ssh" which is a "secure shell connection" using the key "AWSTesting.pem".

PostGIS Installation and Configuration:

Note: Much of this section is adapted from Ihor Tomilenko's Github page 'Setting up PostgreSQL' and is modified to install postGIS rather than postgreSQL alone.

Programs and packages can be installed in Ubuntu using the powerful `apt-get` command. The command works with Ubuntu's "Advanced Packaging Tools" to install, modify, and remove software packages within the instance. This makes installing programs extremely simple once familiar with the terminal interface. The following command can be used to install postGIS, a package containing postgreSQL and the postGIS extension.

```
ubuntu@ip-172-33-22-10:~$ sudo apt-get install postgis

Note: 'ubuntu@ip-172-33-22-10:~$' is just a visual cue to what the user would see in the
command line, the command necessary to install postGIS is the command that follows the '$'
symbol.
```

The word 'sudo' is a command that gives the user the permissions to install the program within the file system as a 'super user,' which is comparable to administrative access in Windows. Next, 'apt-get install postgis' installs postGIS on the AWS instance.
The following command will build and initiate a postgreSQL database.

```
ubuntu@ip-172-33-22-10:~$ sudo service postgresql initdb
ubuntu@ip-172-33-22-10:~$ sudo service postgresql start
```

Once initiated, PostgreSQL must be configured to accept connections from outside IP addresses through a specified port. This is configured through the configuration file 'pg_hba.conf'. To edit a text file in Ubuntu, one can use VIM, which is a inline text editor that can be used right in the terminal window. To initiate VIM, simply type vim in the command line, followed by the path the file you want to edit. One may need to add 'sudo,' if the file you're editing needs special permissions.

```
ubuntu@ip-172-33-22-10:~$ sudo vim /etc/postgresql/9.5/main/pg_hba.conf
```

This will open pg_hba.conf in VIM, to edit a file, press the `i` key to initiate the insert function, and add the following text to the end of the file.

```
# "local" is for Unix domain socket connections only
local   all             all                                     ident
# IPv4 local connections:
host    all             all             ::1/128                 ident
host    all             all             0.0.0.0/0               md5
# IPv6 local connections:
host    all             all             ::/0                    md5
```

Next, the `esc` key will exit insert mode, and `:wq` to write and quit VIM. Adding these lines to the pg_hba.conf file allows for remote users from any IPv4 or IPv6 address to connect to the database. Next, postgreSQL needs to be configured to listen for connections from outside IP addresses. This can be accomplished by editing the postgresql.conf file.

```
ubuntu@ip-172-33-22-10:~$ sudo vim /etc/postgresql/9.5/main/postgresql.conf
```

Find the following line and remove the # sign to uncomment it.

```
#listen_addresses = 'localhost'                 # what IP address(es) to listen on;
```

Replace `'localhost'` with `'*'`. This will tell postgreSQL to listen to all IP address that try to connect to it. Next uncomment the line containing port 2 lines below the previous line. The postgresql.conf file should look like this-

```
listen_addresses = '*'                  # what IP address(es) to listen on;
                                        # (change requires restart)
port = 5432                             # (change requires restart)
```

Edits to the previous two files require postgreSQL to be restarted, in order to reconfigure the settings.

```
ubuntu@ip-172-33-22-10:~$ sudo service postgresql restart
```

The last task necessary to enable remote database access is to create a security group for postgreSQL within the AWS console. On the AWS website, within the ‘Network &

Security' tab of the AWS EC2 web console, 'select create security group', and add an inbound rule for postgreSQL, and select 'anywhere' for the source.

172-31-38-50

Finally, in the instances tab, select Actions > Networking > Change Security Groups, and check off / assign the security group for postgreSQL. Now the postgreSQL database server can be connected to from any IP address, and one can use a postgreSQL GUI like pgAdmin3 or QGIS to edit the database.

Schema Creation:

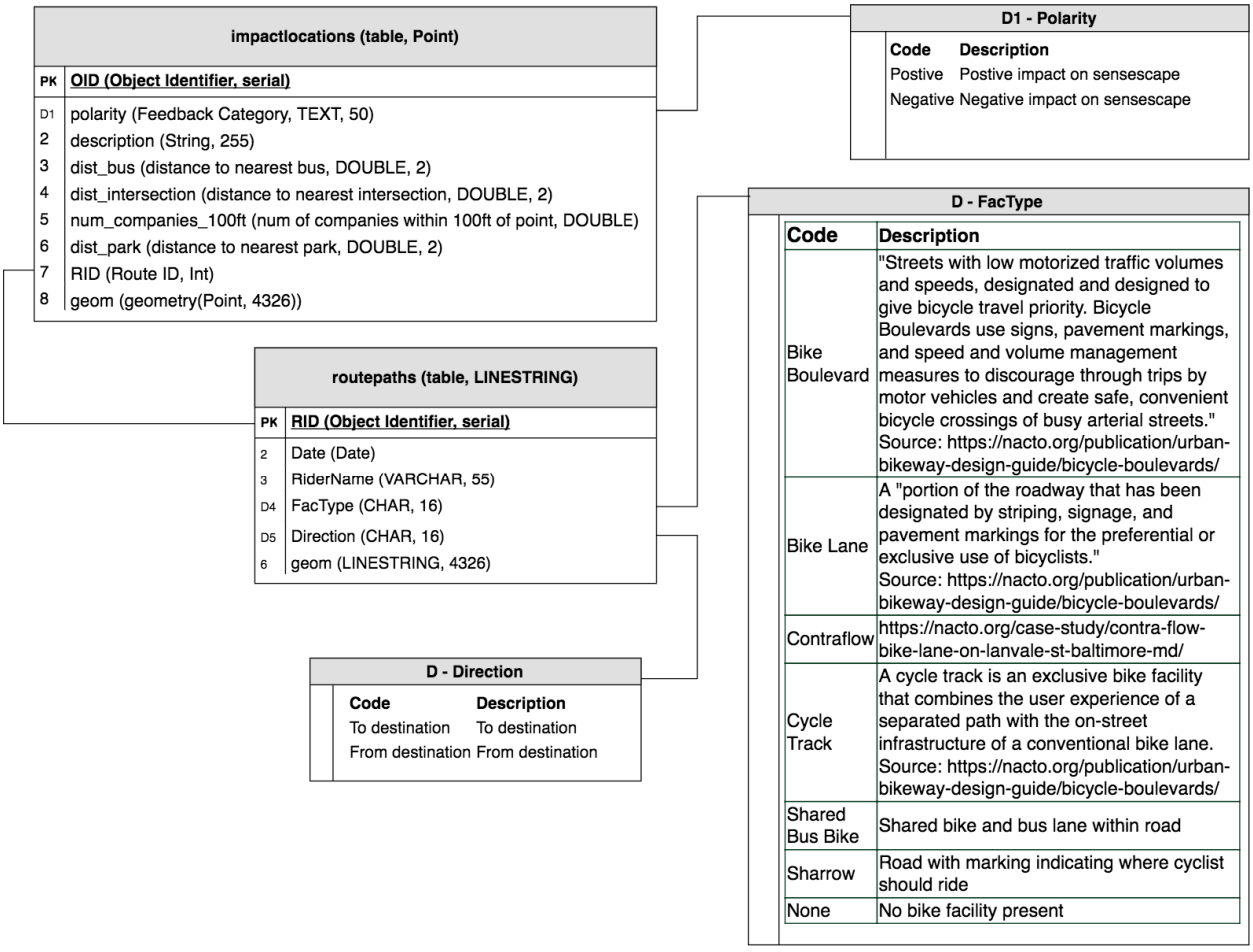
The following commands will create a spatial database within the database server.

```
ubuntu@ip-172-33-22-10:~$ sudo su - postgres
postgres@ip-172-33-22-10:~$ psql
postgres=# create schema bikedb;
postgres=# CREATE user username PASSWORD 'password';
postgres=# GRANT ALL ON SCHEMA bikedb TO username;
postgres=# GRANT ALL ON ALL TABLES IN SCHEMA bikedb TO username;
postgres=# CREATE DATABASE bikedb with owner username;
postgres=# \connect bikedb;
postgres=# CREATE EXTENSION postgis;
```

In summary, the above commands create the schema 'bikedb' and create the user 'username' within that schema with the password 'password'. The user was granted permission on all databases and tables within the schema. The database bikedb was created with the owner of 'username' and the extension postgis was created within the database, which imports all the spatial functions needed.

Tables & Domains:

Bike Sensescape Database



The above diagram summarizes the proposed table schema for the survey database. The database is composed of two tables that share a one to many relationship through the field RID (Route ID). Domains are used to regulate acceptable values for specific fields, and used in the fields polarity, FacType, and Directions. See the table for more information about the fields and domains. A sql file has been included in this report for the creation of these tables.

Schema Discussion & Analysis Potential:

These tables were designed to provide enough data to begin to make predictive inferences into a city's bicycling network. The table 'impactlocations' is categorized by

the polarity field, which determines if the location has a positive or negative impact on the cyclist's ride experience, and then records a number environmental factors that provide inference into what determined the user's selection of polarity. For example, the distance to nearest bus stop (dist_bus) have been shown to increase the probability of a bicyclist having a negative experiences for bicyclists, while propensity to the nearest park (near_park) has been shown to increase the probability of a positive experience (Snizek et al. 2013). These fields can be hidden to the user in the survey form and be post processed using spatial functions within postGIS. The following query would generate the distance to the nearest bus stop.

```
SELECT ST_Distance(g1.geom,g2.geom)
FROM busstops As g1, impactlocations As g2
WHERE impactlocations.oid = 1
ORDER BY ST_Distance(g1.geom,g2.geom) LIMIT 1;
```

This would return the nearest bus stop to the impact location with oid = 1. With further development, a trigger could be placed in the table that would calculate this field on insertion of the point. Point locations can be collected from city open data portals imported into the database using the shp2pgsql function built into postGIS. More environmental fields can be added based on the availability of environmental data within the city the survey is implemented.

With the collection of point locations, their polarity, and the environmental factors that are inherent to the point locations, predictive models can be developed and run to determine the likelihood of a cyclist to have a positive or negative experience at any randomly generated point within a metropolitan area (Snizek et al. 2013). The resulting model could be used by city planners and developers to create smarter transportation network and improve current infrastructure.

Conclusion:

This paper outlines the steps for creating an open-source, distributable spatial database. This database is intended for use in the implementation of a cyclist experience survey within any metropolitan area. The paper includes a recommended schema for data collection. This schema is intended to optimize analysis and deliverable end products from the implementation of the survey. The scope of this project only covers the backend, database side of the survey- but provides a backbone that could be expanded upon through web services and a front end user interface. Future efforts will focus on developing database trigger functions that would auto-populate environmental fields (outlined in the Tables & Domains section), creating web services and web survey form, and creation of regression models that can help decision makers within a city.

References:

Bernhard Snizek, Thomas Alexander Sick Nielsen, Hans Skov-Petersen, Mapping bicyclists' experiences in Copenhagen, Journal of Transport Geography, Volume 30, June 2013, Pages 227-233.

Degen, M, Regenerating Public Life? A Sensory Analysis of Regenerated Public Spaces in El Raval, Barcelona. In: Hinchcliffe, D., Rugg, D. (Eds.), Recoveries and Reclamations. Intellect Books, 2002, Pages 19–35.

Jan van Duppen, Bas Spierings, Retracing trajectories: the embodied experience of cycling, urban sensescapes and the commute between 'neighbourhood' and 'city' in Utrecht, NL, Journal of Transport Geography, Volume 30, June 2013, Pages 234-243.

Ihor Tomilenko, Setting up PostgreSQL, Github, April 28, 2017. https://github.com/snowplow/snowplow/wiki/Setting-up-PostgreSQL

Tight, M., Timms, P., Banister, D., Bowmaker, J., Copas, J., Day, A., Drinkwater, D., Givoni, M., Gühnemann, A., Lawler, M., Macmillen, J., Miles, A., Moore, N., Newton, R., Ngoduy, D., Ormerod, M., O'Sullivan, M., Watling, D, Visions for a walking and cycling focussed urban transport system, Journal of Transport Geography 19, 2011, Pages 1580–1589.